IBM

General Information Manual

IBM

General Information Manual

7030 Data Processing System

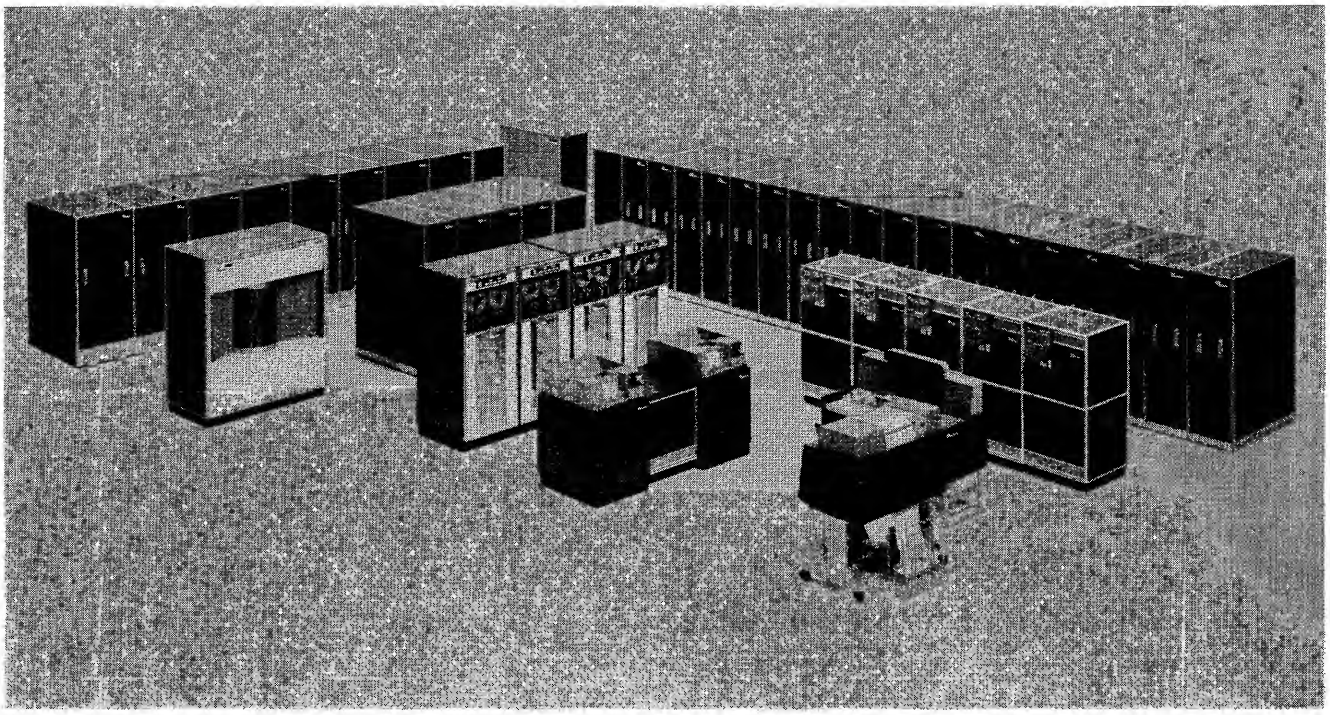
Preface

Modern civilization—with its finance, commerce, industry, science, and business—would be impossible without calculations. From the time man found a need for counting, the problems of adding, subtracting, and recording the results have grown steadily. To cope with these problems, many devices have been developed. Modern computers, electronic data processing systems, have become invaluable to scientist and businessman alike.

Data processing consists of planned actions and operations upon data to produce a desired result. Data processing systems find ever-widening applications in solving new and larger problems in every phase of science, business, and industry. Rapidly expanding scientific investigations involve billions of complex calculations; increasing amounts of data are required for control in industry, business, and government. The result is a demand for data processing systems with vastly improved performance. The IBM 7030 Data Processing System is such a system.

This manual presents the operating principles of the new IBM 7030. The manual is not intended for detail work but to present an over-all approach to the system.

Contents



IBM 7030 Data Processing System

IBM 7030 Data Processing System

The IBM 7030 Data Processing System is a high-speed, general purpose data processing system that substantially exceeds the over-all performance of its predecessors in both technical computing and business data processing. Large storage capacity and microsecond computing speeds allow increased efficiency on present problems, combining small runs into more efficient larger problems, and solving problems too vast to be practical on lesser systems.

An important component of the 7030 data processing system is the IBM 7302 Core Storage with a capacity of up to 16,384 words. Multiple storage units may be attached to the system and cycled independently to make several storage references simultaneously and to achieve an effective data rate much faster than the basic cycle. Space is provided in the instructions to address 262,144 words.

A 7030 data processing system may also comprise multiple computers, either sharing the same storage or coupled more loosely through their IBM 7619 Exchange units. In this way, the computing efficiency of the system may be greatly expanded if the need arises.

The 7030 system combines in one computer system many desirable features found only independently in its predecessors: high-speed binary fixed point and floating point arithmetic; decimal and binary arithmetic performed on fields of variable length; and manipulation of data in virtually any desired character representation. Thus, the 7030 system may be used to solve an unusually wide range of problems in all data processing fields.

The instructions developed for this system exhibit many new and powerful features. Fewer instructions are usually required to write a given program. Most instructions are half-word size. Fewer and shorter instructions mean less storage space required for programs, fewer references to storage, and fewer instructions to be executed. These factors contribute to the high performance of the IBM 7030 Data Processing System.

Most instructions may have their address part modified by automatic indexing in a separate index arithmetic unit. Some instructions may be executed completely within this unit while other arithmetic operations are being executed in the main arithmetic unit. This overlapping of instruction execution allows a significant increase in over-all system performance. The computer automatically adjusts the flow of data between the input-output units, instruction processor, arithmetic processor, and core storage despite the complexities of time sharing.

Emphasis is placed on efficient floating point, indexing, and branching instructions (the core of many programs in the technical field) and on data processing on fields of varying length regardless of the word boundaries. A set of instructions is available to perform decimal or binary arithmetic as well as radix conversion and logical operations on these variable length fields. These instructions, coupled with indexing and branching, offer a powerful tool for all phases of business and commercial work. They may also be applied to problems common to business and technical fields, such as program preparation and editing of input-output data.

The input-output system is flexible; almost any device furnishing or accepting digital data may be connected to the system. The IBM 7619 Exchange controls all input-output devices and transmits data between these units and core storage while computing proceeds in the IBM 7101 Central Processing Unit. An IBM 7612 Disk Synchronizer, capable of controlling up to 32 disk storage units, may parallel the operation of the exchange and transmit data to and receive data from core storage at the same time as other input-output devices.

Some devices that may be attached to the 7619 exchange are: card readers, card punches, printers, magnetic tape units, consoles, and console printers. The exchange can control up to 32 data channels that operate concurrently or independently, up to a maxi-

mum data transmission rate of 100,000 words per second between the exchange and core storage.

Automatic error detection and correction, automatic means of localizing faults, and other maintenance aids reduce the time required to identify and correct any machine fault which may arise. All data transmission is automatically checked for validity by a unique combination of error check and correct bits that is carried with each word. Any single bit error within the word being transmitted is automatically corrected. All double and most multiple bit errors are indicated for program interruption. In all other areas of the system, extensive checking is built in. This automatic checking provides continuous monitoring to insure system reliability.

Features that permit improved operating techniques also contribute to performance. Routine operating functions are, wherever possible, placed under direct control of the stored program. Input-output units require a minimum of manual setup. Control panels are omitted entirely; all arrangement and control of data is done by programming. Multiple input-output channels, buffering, and multiprogramming facilities make possible efficient conversion, code translation, and editing on independent input-output units attached to the system, with virtually no increase in major program execution time. Physical switching of units or manual transfer of tape reels is substantially reduced.

A new concept is applied to the operator's console. Instead of being intimately associated with the central processing unit, the console is separated from the main computer and becomes an optional input-output device. Keys, switches, lights, digital display, and the console printer are all subject to programmed interpretation and control. This interpretive approach and an extensive, flexible program interrupt system provide facilities for close communication between man and computer when human intervention is desired. These features are provided because intelligent human intervention and supervision can often bring a problem to completion more quickly than computation alone.

All console features are designed so that advantage may be taken of the intervention techniques made possible by multiprogramming (having a number of problems available to the computer so that necessary time waits in each problem are utilized by operations on one of the other problems). When multiprogramming is used, the high-speed computer is not required to wait for the thinking and reaction time of its user. The economics of human intervention are radically changed; the computer can go on with useful work while the user ponders his next move.

System Organization

The 7030 system includes a central processing unit, storage bus control, core storage, exchange, input-output devices, and disk storage system.

To achieve maximum use of all components, many computer sections can operate at the same time. Each core storage unit, the instruction processor, look-ahead, arithmetic and logical units, and each exchange channel is independent. Two or more of any of these units (depending on the particular problem) can operate concurrently and autonomously.

The 7030 system is capable of overlapping input operations, output operations, and operations in the central processing unit. The system can also overlap the actual processing and execution of instructions themselves.

Data Flow

Figure 1 shows the main sections of the IBM 7030 Data Processing System. Information moves between the input-output devices and core storage under control of the exchange. The storage bus control acts like a telephone switchboard to actually route the data. The central processing unit consists of registers, arithmetic units, and control circuits necessary to operate on data taken from storage; the unit is controlled by instructions also taken from core storage.

The following data fetching operation (Floating Add) illustrates data flow and the instruction path through the system:

1. An instruction counter requests the next instruction from storage by sending the location of the instruction to the storage bus control.

2. The storage bus control decodes the address and requests the instruction from the storage unit.

3. The instruction is sent from storage to the instruction processor where two identical, full-word registers store up to four half-word instructions while the instructions are being modified. Instruction fetches (getting an instruction from storage) normally alternate between these registers.

4. The instruction is sent through an Error Check and Correct (ECC) check point; while the instruction is checked, it is also decoded.

5. If the instruction is to be indexed, the proper index word is combined with the address field of the instruction to obtain an effective address.

6. If the contents of the effective address are not held in look-ahead, a data fetch operation (getting

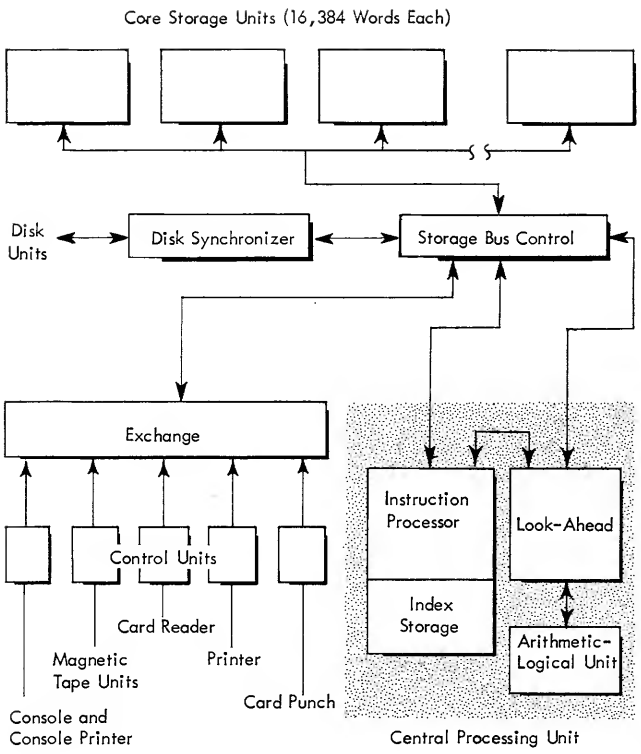


Figure 1. IBM 7030 Data Processing System, Data Flow

data from storage) is started by sending the effective address to the storage bus control.

7. Before the data fetch is started, the instruction is loaded into one of four look-ahead levels; these levels are identical logically and contain the instructions to be executed and their data.

8. The appropriate indicators and the instruction counter contents are also loaded into a look-ahead position; in case of an interruption, enough information is always present to allow the computer to automatically recover data for a restart operation.

9. When the data requested by look-ahead return from storage, they pass through an ECC check point and at the same time are loaded into their corresponding look-ahead level.

10. The instruction processor, meanwhile, has proceeded to the next instruction in sequence so computer time is not wasted.

11. When the instruction is to be executed, its data are loaded into a register in the arithmetic unit.

12. The data and the accumulator contents are combined in the adders.

13. The result returns to the accumulator and is again validity checked.

Other instructions follow slightly different procedures but all follow the rule that instructions are handled logically as if they were being executed sequentially.

Core Storage

The 7030 system is equipped with high-speed core storage units as primary storage. The IBM 7302 Core Storage is completely solid state and of modular construction to maintain the high reliability and compactness of the system.

The 7302 core storage is subdivided into 16,384 full word locations. Each location contains 64 information bits and 8 error check and correct bits. As shown in Figure 1, a 7030 system may use multiple core storage units. Multiple units may be used, for a possible 262,144 full-word locations. Any possible word location may be expressed in the 18-bit location address portion of an instruction. Six additional bits are used to address any bit in any word (Figure 2).

A complete write operation (putting new data into storage) or read operation (reading out data and replacing it) is accomplished in a full core storage cycle of only 2.18 microseconds. Each storage unit operates independently and problem solution time may be significantly decreased by adding additional core storage units to the system. This increased performance, due to the overlapping that occurs in the various references to core storage, is in addition to the usual advantages of larger storage capacity, such as the ability to completely contain a large program in storage at one time. To take full advantage of overlapping, successive addresses are cycled among the core storage units. With four storage units, it is possible to approach an effective storage cycle of .5 microseconds.

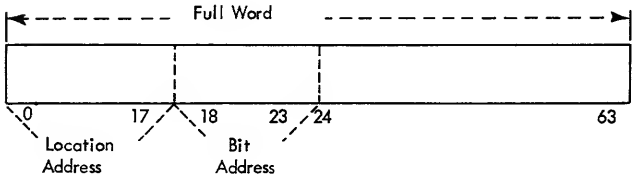


Figure 2. Instruction Address Bit Format

Information Format

One of the design goals for the IBM 7030 system is that it perform equally well on large technical, business, computational, and logical problems. Formerly, computer design differed greatly, depending on whether the computers were principally aimed at technical or business applications. Experience has shown that each design is restrictive, that these two areas do not exhaust the field, and that the computing load of a single installation rarely falls into just one such class.

Examination of data handling within computers and of the fundamental units of data in different kinds of data processing tasks led to a new organization which contributes greatly to the power and flexibility of the 7030. In addition to a 96-bit parallel arithmetic unit of very high speed and a 24-bit parallel unit for address arithmetic, the 7030 system has a serial unit which can handle up to 8-bits in parallel for processing variable length data.

sign and exponent flag, fraction with sign, and three data flags. Floating point operations must proceed with utmost speed, and a fixed format facilitates parallel arithmetic. This format corresponds to the capacity of a storage location with a length of 64-bit positions. The word includes a 1-bit exponent flag, 10-bit exponent, 1-bit exponent sign, 48-bit fraction, and 4-bit sign byte. (A byte is 1 to 8 bits that are treated as a unit.) The three flag bits of the fraction sign byte serve the entire word. Figure 3 shows the data unit for floating point arithmetic.

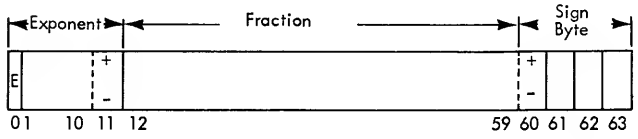


Figure 3. Floating Point Word Format

Data Lengths

Five common types of operations may be associated with automatic data processing tasks: floating point arithmetic, fixed point arithmetic, address arithmetic, logical manipulation, and editing operations. Each operation uses data differing in length and internal structure. An ideal computer would permit each operation to address the required data length directly and simply by using all properties of data that are constant.

To handle data that differ in structure and length, the 7030 system provides the information about data length in the instruction that uses the data. This provides a logical and flexible means of treating different data lengths without loss of core storage capacity or additional instructions. Each instruction that can refer to variable length data contains the complete address of the left-most (high-order) bit of the field and the length of that field. Instructions that do not refer to variable data lengths are not burdened with defining them.

Floating Point Data Length

The data length for most technical computation is the floating point number because floating point operations free the programmer from all details of scaling analysis and decimal point alignment. This data format has a rigid internal structure: the representation of a single number includes exponent with

Fixed Point Data Length

Fixed point arithmetic is used on problem data where scaling analysis is negligible, such as data encountered in business or statistical calculations. Numbers may or may not be signed (Figure 4). If the arithmetic is binary, the data have a structure of digits which are individually coded into binary representations. Whether the data have a simple structure or a complex one, the natural length is variable; typical numbers vary from four to forty bits in length.

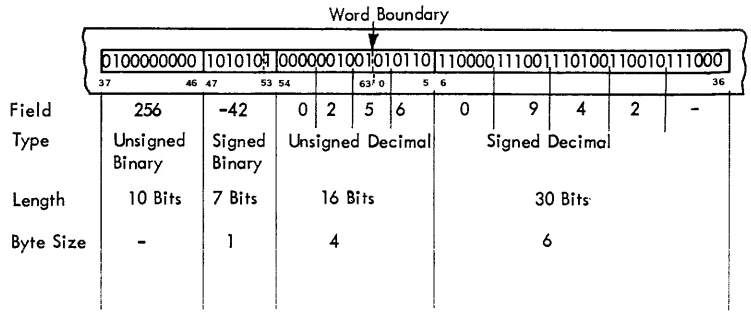


Figure 4. Data Types, 7030 System

Address Arithmetic Data Length

Address arithmetic operates upon data whose structure is similar to that of fixed-point data, whether decimal or binary. Figure 5 shows the three major instruction formats in the 7030.

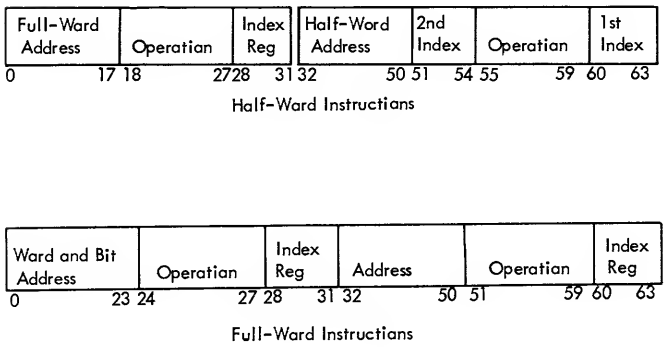


Figure 5. Instruction Formats

Logical Manipulations Data Length

Logical manipulations—whether used as a main part of a program as in data analysis or used to control the course of the program—operate on a data structure composed of a group of bits, each of which has an independent meaning. This distinguishes such operations from arithmetic operations which use bits as components of numbers. Each bit in a logical field may represent parts of a mask, contents of certain indicators, or other information used to obtain a logical result. Figure 6 shows two logical fields, from different words, which are to be combined and then stored in a third field in another word. In the operation shown, each field position of the word 1000 is combined with the corresponding field position in word 1001. For each position in the result, the logical operation places a 1-bit when word 1000 had a 1 and word 1001 had a 0 for that particular position. All other bit combinations result in a 0.

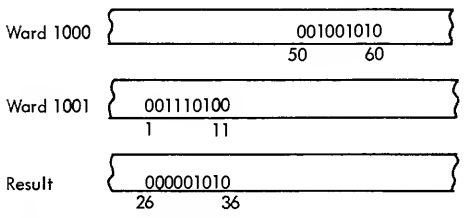


Figure 6. Data Fields for Logical Manipulations

Editing Operations Data Length

Editing operations include all operations in which data are transformed from one format to another, checked for consistency with a source format, or tested for controlling the course of the program. Data for such operations vary widely (Figure 7). All data of the previous types of operations are subject to and undergo certain editing operations in the normal course of their processing. There are, however, other data fields that are unique to editing operations. For example, a group of data fields may be moved as a unit within main storage to assemble records for output.

Editing operations possess not only the most complex data structures but also the most widely varying field lengths. For some manipulations, field length is a single character; for others, a field is many characters.

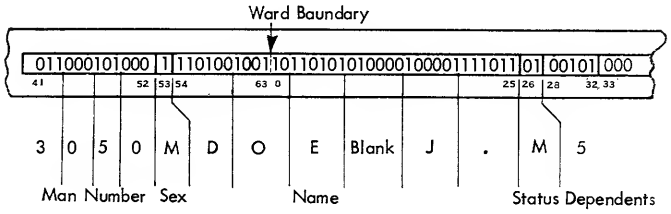


Figure 7. Data Fields for Editing Operations

Variable Field Length

To obtain a high data rate from core storage, it is necessary to define words having a large number of bits which are read or written in parallel. The word length is set at 64 information bits, plus 8 error check and correct bits. This length permits word addresses and bit addresses to be manipulated or indexed with a single arithmetic operation. Carries from binary operations on bit addresses are converted to proper word addresses.

To be practical, floating point number representations should be longer than 32 bits; a length of 64 bits is ample. With 64-bit floating point numbers, multiple precision operations may usually be avoided, improving performance on this type of problem.

The full word (64 bits) is also a convenient size for instructions that must specify a variable length

field. Instructions operating on the more rigid formats of floating point and address arithmetic do not need variable field specifications and are abbreviated to 32-bit size. Providing both full-length and half-length instructions leads to more efficient use of instruction bits and substantially improves computer performance. A full-length instruction is not restricted to a single word but may occupy adjacent halves of consecutive words. Thus, no wasted core storage positions occur when full-word and half-word instructions are intermixed.

The maximum size of directly addressable core storage is 262,144 words. Addressing to the bit level requires 24 address bits: 18 high-order bits to specify word location and 6 low-order bits to refer to one bit within the word.

For operations on variable length fields, it is usually necessary to specify the inner structure of the field. For alphabetic fields, this consists of the individual letters or other characters. For numerical fields, the structure includes the sign and the digits. These subunits are called bytes. Because the coded representation of a byte varies in size, byte sizes of one to eight bits may be specified as shown in Figure 7.

Instruction Processing

In most stored program computer systems, the execution of an indexed, data fetching instruction goes through the following sequential operations:

1. Advance the instruction counter.
2. Send a request to main storage for the instruction.
3. Main storage sends the instruction back.
4. Decode the instruction.
5. Perform index arithmetic, if required.
6. Send fetch request to main storage for the operand.
7. Main storage sends the operand back.
8. Perform the arithmetic operation.

Each of these steps takes time, but only the last item represents "useful" work. The other steps, although necessary, should be considered as preparatory operations.

The 7030 system uses two sections of the central processing unit, called the instruction processor and the look-ahead, to accomplish this preparatory work. The instruction processor has an independent indexing arithmetic unit. This unit prepares the effective addresses of instructions and initiates storage references to core storage units simultaneously with the execution of preceding instructions. Fetched data are held in look-ahead until needed by the arithmetic and logical unit. This organization has two desirable purposes:

1. Most preparatory operations are done in parallel and do not delay the main calculations.
2. Several core storage units can operate simultaneously, giving the effect of faster core storage speed.

An additional benefit is realized in instruction execution time. Using the instruction processor and look-ahead, the 7030 is able to differentiate between the processing of housekeeping (program maintenance) instructions and data-handling (problem solving) instructions. Housekeeping instructions can often be executed directly in the instruction processor with practically no execution time delay to the system. For data-handling instructions, effective addresses are already formed and data fetched from storage, so minimum time in the arithmetic unit is required for their execution.

Despite internal complexities arising from overlapping instruction and data fetches, address modification, and the actual execution of instructions, no timing restraint is placed on the programmer. The only difference in system operation due to these features is a higher program execution speed. Look-ahead takes care of the unusual timing conditions caused by overlapping and makes the program appear to be executed one instruction at a time, just as it is written.

Arithmetic Unit

The arithmetic unit consists basically of two sets of circuitry. One set performs high-speed parallel floating point arithmetic on full words. The other set performs serial arithmetic and logical operations on fields of variable length; bits may be operated on as units or as numbers coded in decimal or binary format. The adder section of the arithmetic unit adjusts its mode of operation to the type of arithmetic specified by the instruction.

For simplicity, the arithmetic unit may be thought of as four one-word registers and a short register (Figure 8).

Registers A and B constitute the left and right halves of the accumulator register and are directly addressable as locations 8 and 9 when programming. Registers C and D serve as storage registers, receiving words from storage and assembling results to be stored. Registers C and D always hold copies of the contents of core storage words and are not addressable. The S register, addressable as location 10, stores

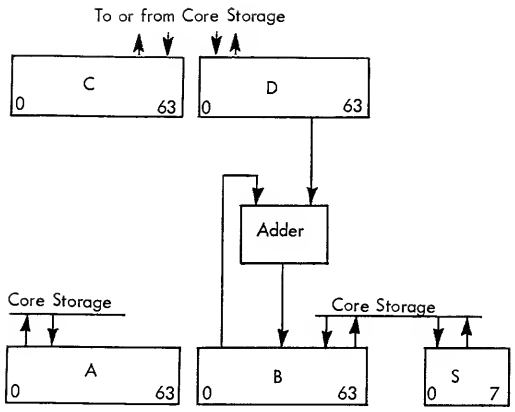


Figure 8. Arithmetic and Logical Section, Unit Registers

the accumulator sign bit, data flag bits, and zone bits; these bits, collectively, are called the accumulator sign byte.

In floating point addition, an operand from storage may be considered to be in register C. It is then added to the contents of register A and the result is placed back in register A. When using the built-in double precision facilities, register B would be coupled with register A to expand its capacity.

In floating point multiplication, one factor comes from core storage to register C, and the other is taken from register A to register D. The product is developed in the cleared accumulator. In cumulative multiplication, one factor comes from storage to register C while the other comes from a factor register (not shown in figure 8) to register D. The two factors are multiplied and added to the accumulator contents.

In variable field length operations, the storage word or words containing the operand are placed in C and D. The operand is selected a few bits at a time and processed. In some operations, the result replaces the accumulator operand; in others, the result replaces the storage operand; in others, only certain indicators are affected, and both operands remain unchanged. Binary integer multiplication and division operands are stepped into the parallel circuitry a few bits at a time, but the actual operation is performed in parallel.

In division the quotient appears in the accumulator and the remainder is developed in C and D. At the conclusion of the divide operation, the remainder returns to the remainder register. The remainder and factor registers are assigned core storage locations 13 and 14 respectively.

Operations

The operations available in the 7030 system may be divided into the following broad categories:

Floating Point Arithmetic
Integer Arithmetic
Radix Conversion
Index Arithmetic
Branching
Data Transmission
Input-Output

The arithmetic instruction set includes the elementary operations load, add, store, multiply, and divide. Modifier bits are available to manipulate the operand sign. Thus, the operations subtract or add absolute are obtained by use of sign modifiers with the add instruction and are not provided as separate operations. The sign modifiers also permit changing the sign of a number to be loaded, stored, multiplied or divided.

A feature of the multiply operation is that one of the factors is taken from the accumulator rather than a separate register. Thus, this factor may be the result of previous computation. Similarly, divide places the quotient in the accumulator and makes it available for further arithmetic steps.

Extensions of the basic set of arithmetic operations permit adding or counting in storage, rounding, cumulative multiplication, comparison, and other variations of the standard add operation. One variation is an add-type operation called add magnitude. This operation is like add when numbers of like signs are involved. With numbers of unlike signs, the operation is different from a subtraction (adding with unlike signs) in that it does not allow the sign to change. Instead, the result is set to zero. This operation is useful when dealing with non-negative numbers.

Most arithmetic operations are available in the floating point mode as well as in the fixed point or integer mode. The floating point set includes additional instructions to handle portions of a floating point number and double length numbers. A floating point square root operation is also provided.

Floating Point Arithmetic

Most specifications of floating point arithmetic have been explained. In summary: floating point arithmetic is done in binary using a specialized data format which occupies a full 64-bit word. The emphasis is on high-speed computation of large mathmetical

problems. The binary radix makes available techniques which greatly speed multiplication and division.

The 48-bit fraction and the storage efficiency of the binary system make it possible to compute in the single precision mode a large number of problems that were done in the multiple precision mode on other systems. When required, however, direct multiple precision mode may be programmed using the floating point instructions.

Floating point arithmetic may be performed in either a normalized or unnormalized mode. The latter mode also provides a method of doing fixed-point binary arithmetic.

Integer Arithmetic

Integer arithmetic encompasses all data arithmetic on other than specialized floating point numbers. The emphasis is on versatility and economy of storage. The integer arithmetic instruction set is similar to the floating point set; most operations have the same names and analogous meanings.

Integer arithmetic may be performed directly in either decimal or binary radix. Individual numbers or fields may be of any length, from 1 to 64 bits. Fields of different lengths may be assigned to adjacent locations in storage, even if a field lies partly in one storage word and partly in the next. Each field may be addressed directly by specifying its position and length in the instruction; the computer takes care of selecting the storage words required and alters only the desired information.

Individual characters or bytes in a field may also be varied in length. Thus, a decimal digit may be compactly represented by a binary code of 4 bits, or it may be expanded to 6 or more bits when intermixed with alphabetic information. Decimal arithmetic may be performed directly on a decimal number, regardless of how many bits are used to code a digit. Because decimal digits, alphabetic characters, and other single symbols may be coded in several ways, the term byte is used to denote a single group of bits processed together. A field may consist of one or more bytes. Considerable latitude may be used to specify the coding of alphabetic data through use of zone bits. The coding may conform to the 6-bit binary coded decimal representation, as used in the IBM 705 Data Processing System, or other convenient representations may be used.

The name integer arithmetic is derived from the fact that in division the result is normally aligned as if the operands were integers. It is possible, however, to specify that operands be offset to obtain any desired alignment of the radix point. An offset may be specified in every instruction; there is no need for separate instructions to shift the contents of the accumulator. Numerical data may be signed or unsigned. For unsigned data, the sign is omitted in storage, saving space and avoiding the task of assigning signs where there were none. Unsigned numbers are treated arithmetically as if they were positive.

A significant feature of integer divide is that it will produce meaningful results regardless of the magnitude of the dividend or divisor (provided the numbers are of proper register size). The only exception is a zero divisor which will turn on appropriate flag indicators. This feature eliminates much of the scaling previously required before a divide instruction could be accepted.

Alphabetic and nonnumerical fields of various length may be handled by integer arithmetic operations as if they were unsigned binary numbers, regardless of the character code or the number of bits used for each character; there is no fixed character code built into the computer. Alphameric high-low comparisons are made by a binary subtraction of two fields. The only requirement is that the binary numbers representing each character fall into the comparing sequence desired for the application. Another use of integer arithmetic operations is to perform general arithmetic on portions of floating point words, instruction words, or index words.

All integer arithmetic operations are made available in either decimal or binary form by setting one modifier bit. Decimal multiplication and division are not built into the computer directly; their operation codes are used to cause automatic entry into a subroutine that takes advantage of high-speed radix conversion and binary multiplication or division. Decimal multiplication and division are thus as convenient to program as if they were built in.

Radix Conversion

Radix conversion operations provide for decimal input-output while retaining the advantages of binary operation within the computer. These operations are also used in the decimal multiplication and division subroutines mentioned in the preceding section.

Several operations are provided to allow a variety of locations for the operand and the result. A field from storage may be converted and placed in either the accumulator or a transit register. Alternatively, a field from the accumulator may be converted and the result returned to it. In these operations the operand is an integer and may be converted from binary to decimal or from decimal to binary format.

Connectives

Instructions which logically combine bits by AND, OR, and EXCLUSIVE OR are included in the connective instruction set. This set also provides logical facilities not previously available.

The connective operations are called connect, connect to storage, and connect for test. Each connective operation specifies a storage field of any length from 1 to 64 bits, as in integer arithmetic. Each bit in the storage field is logically combined with a corresponding bit in the accumulator. There are 16 possible ways to combine, or connect, two bits. Each of these connectives may be specified with each of the three connective operations. Besides the connectives AND, OR, and EXCLUSIVE OR, there are connectives to match bits, to replace bits, to set bits to zero or one, and to invert either or both operands.

Two counts describing the result field are available after a connective operation. The 7-bit all ones counter contains a count of the ones in the result field. The 7-bit left zeros counter contains a count of the number of zeros to the left of the most significant 1-bit in the result field. These counts are positioned so that they can be readily used for indexing.

These logical operations are neither modifications of arithmetic nor auxiliaries to it, but are equal to arithmetic in importance. The logical operations combined with the variable field length mechanism permit conversion of data formats and analysis of the resulting data and constitute a complete and powerful system for operating on groups of independent bits rather than on numbers.

Index Arithmetic

Index functions may be divided into four groups: address modification, index arithmetic, termination, and initializing. The first group is used to address operands and justifies the existence of index quantities. The other groups concern changes of index quantities, tests for end conditions, and set up procedures; these operations are often called housekeeping. All index arithmetic is performed in a separate index arithmetic unit. Many of the housekeeping operations are also executed directly in this unit.

The address part of most instructions may be modified by adding a number in a specified index register before using the address. Normally, both the instruction and the index register remain unchanged. Altering index registers is the function of the index arithmetic operations. The instruction set includes operations for loading, storing, incrementing, and comparing index values. The index value is a signed number and the additions are algebraic. One of the instructions allows up to 16 index values to be added together for use in further indexing. Another indexing instruction provides the function of indirect addressing.

Each time an index word is altered by index arithmetic, a test may be performed to determine when the last element of the array is addressed. This process is called termination. The form of test used in the 7030 system allows the test to be independent of the base address and increment, so that even an increment of zero is permissible. This is done by reducing a count field by one every time a loop is traversed. Each index word contains a count field, and counting may be coupled with incrementing the index value. A third field in each index word specifies a refill address with which another index word may be loaded automatically. Together these three fields provide a convenient indexing technique. The index word format is shown in Figure 9. At each traversal of a program loop, a signed increment is added to the index value and the count stepped down by one. When the count reaches zero, the index register is reset by refilling it from the storage location which contained the original value and count. All this may be done with one indexing instruction at the appropriate point in the program loop.

The instruction set permits many other indexing techniques. An important one is the use of the refill address to indicate the next index word in succession in an indexing chain. Such chains permit the computation to progress through a series of items or records which are not stored in the order in which they are to be used. Chaining can greatly simplify insertion, deletion, and sorting of items by not requiring rearrangement of the data in storage.

Sample Program

The following example, multiply vectors A and B, illustrates the use of count and refill in a technical computation. Each vector has (n) elements. Vector A has its first element at (A_0), vector B has its first

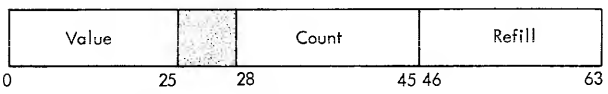


Figure 9. Index Word Format

```
Location    Instruction          Comments
START       LX, $ X2, EYEO       Lood index register 2 from index word EYEO
            LX, $ X3, EYEO       Lood index register 3 from index word EYEO
            L, TRUZERO           Lood accumulator with true zero value
LOOP        LFT(N), AO($ X2)     Load cumulotive multiplicond from AO, indexed by reg 2
            *+(N), BO($ X3)      Multiply cumulotively by BO, indexed by register 3
            V+ICR, $X3,P         Increment reg 3 by P; count ond refill when C=0·
            CBR+,$X2,LOOP        Increment reg 2 by 1; count ond refill when C=0
                                 branch to LOOP when count is not zero
            ST(N), CO            Store cumulative product at CO
EYEO        XW,O,N,EYEO          Index word
TRUZERO     DD(U),0E-610         Constant true volue
```

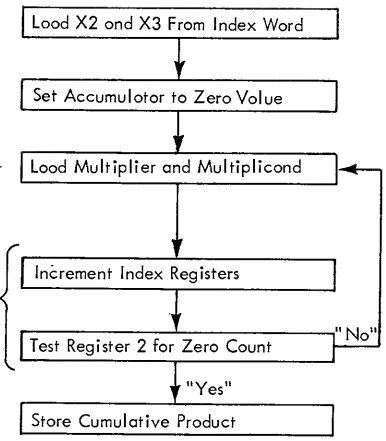


Figure 10. Vector Multiplication, Sample Problem

element at (B_0). The product is to be stored at (C_0). A is stored in successive storage locations while B is a column vector of a matrix whose rows have (p) elements and are also stored successively. Therefore, the elements of B have locations which are (p) apart in storage.

The problem is shown in both flow chart and Strap program coding in Figure 10. (A complete list of all 7030 instructions with their symbols is included in the Appendix.) The product is added to the contents of the accumulator that contains the sum of the previous products. This operation is called cumulative multiplication. When the count in the index registers 2 and 3 goes to zero, they are automatically refilled from the contents of the storage location specified by their address (refill field). The count in index word 2 also controls the termination of the cumulative multiplication.

Instructions generally specify one of a set of 15 index registers for address modification; the number of available registers may be supplemented by other index word locations in storage by a rename operation. This operation identifies one index register with one storage location and does the work necessary to cause this storage location to reflect changes in the index register. While indexing instructions are provided primarily to change index values and counts, it is possible to use these instructions in a manner by which the index quantities may be advanced each time they are used. This mode may be used advantageously to step along a string of data of various lengths without requiring a separate incrementing instruction at each step.

Control Words

Control words specify an area in core storage, the number of words to be transferred, and a refill address. This information is used by the exchange. Control word format (Figure 11) is identical to index word format; a control word may be used, therefore, for both input-output operations and for indexing functions.

Several flag bits are available to control input-output operations. These flag bits have corresponding usefulness in indexing operations. For example, the chain flag in the control word occupies the same position as the index flag in an index word. Thus, it can be used to automatically signal the last section of data during reading or writing and, when the control word is used as an index word, the flag may signal the end of data by an appropriate branch instruction.

Data Transmission

Data transmission instructions transmit data from one storage location to another. One word or a group of words may be transmitted in a single operation. The number of words to be transmitted is specified by a

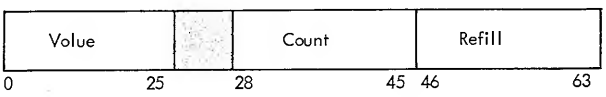


Figure 11. Control Word Format

count field; full core storage capacity may be transmitted in one operation.

Data transmission operations may act in two ways.

When specified by the transmit instruction, a data transmission instruction moves data from one storage location to another. After this operation, the data in the original storage locations remain unchanged. The data that were in the receiving storage locations are destroyed.

When specified by the swap instruction, a data transmission instruction interchanges data. Both sets of data are preserved, but they have interchanged their storage locations.

Branching

Branching operations either conditionally or unconditionally alter the instruction counter to change the course of a program. The number of branching instructions is not large, but modifiers provide great flexibility. A modifier specifies if branching is to occur when an indicator is either on or off. Another modifier may cause the tested indicator to be reset to zero. A second operation, branch on bit, permits testing a single bit anywhere in storage. The tested bit may also be modified. This instruction places an almost unlimited number of indicators under the direct control of the program.

All computer indicators such as: sign, overflow, error, and input-output conditions, are collected in one 64-bit indicator register. The branch on indicator instruction may then specify any one of these 64 indicators as the condition to be tested.

Program Interrupt System

The 7030 program interrupt system permits the computer to respond rapidly to extra or multiprogram demands which occur at arbitrary times. These interrupts are most often signals from the exchange that some external signal has been received or that an input-output operation has been completed. The computer may also make rapid selection of an alternate sequence of instructions when program activated indicators signal that special circumstances have occurred.

The interrupt system uses a continuously monitored indicator register. When an indicator comes on, the computer selects an instruction from a corresponding location in a table of correction instructions. This instruction is sandwiched into the program being executed when the interrupt occurs. Means are provided to select which indicators may cause interrupts and when the interrupt will be permitted. Priorities may thus be established. If more than one interrupt signal occurs at the same time, the system accepts them in the order of their priority.

The desirability of having the interrupt system monitor exceptional conditions becomes apparent when the problem of determining arithmetic overflow is considered. Formerly, it was necessary to program a test of every arithmetic operation whose execution could result in an overflow condition. The alternative was usually a costly machine stop or useless processing with inaccurate data.

In the IBM 7030 system, the programmer can select indicators that are monitored automatically at no loss of program execution time. If overflow or underflow conditions arise, a correction routine may be entered and the condition corrected or, if this is impossible, the processing halted without meaningless computing.

With a high performance computer system, wasted seconds, or even microseconds, are costly. Such waste occurs when the computer asks for a reaction or indication from a control system or an operator. Another source of wasted time occurs during any input-output operation. The best way to avoid these wasted seconds is through multiprogramming – having a number of problems available to the computer so that the necessary time waits in each problem are utilized by operations on one of the other problems. The main requirement for satisfactory multiprogramming is an efficient interrupt system. The interrupt system is also necessary for efficient operation of an independent input-output system, such as the 7619 exchange.

Another feature of importance for multiprogramming is address protection, which permits areas of storage used in one problem to be protected from another problem. Thus, if the computer is solving several problems at the same time, each one may be protected from change by any of the others. The high and low limits of the storage area being used by the program are monitored by indicators. If another program attempts to store new data within the limits of the first program, an interrupt occurs because the original data would be destroyed by this type of operation. However, a data fetch operation may be allowed (from within the limited storage area) because the stored data would not be destroyed by the fetch. This is at the programmer's discretion.

The programmed approach to multiprogramming is used to maintain the flexibility of the system. The programmer has the ability to allocate all facilities freely; if one program needs all of core storage, it may have it. Only by giving the programmer this ability to control computer facilities, can he obtain full use of the computer's power.

Exchange and Input-Output Units

IBM 7619 Exchange

The IBM 7619 Exchange directs information between input-output units, other external units, and core storage. The exchange enables data processing and input-output operations to proceed simultaneously.

The 7619 exchange provides a means of connecting many different units to the computing system. System organization is for input-output units physically near the computer as well as for direct data input from remote sources, output to remote stations, and communication with other computers which may or may not be a direct part of the system. A common method of program control applies to all units (Figure 12).

The 7619 exchange contains common control facilities for input-output and external units. This time sharing of controls keeps the input-output and external units as simple as possible for the programming effort, yet maintains fully overlapped operation. The exchange also does address housekeeping and assembles or disassembles information without tying up computer or core storage time. The only computer time involved is that needed to start and interlock the operation. The only core storage cycles required during external operations are those needed to transfer full words of data to or from locations in core storage. These cycles are fitted between computing operations without interfering with the computer program, except for a possible slight delay for the actual storage cycles.

The basic exchange configuration provides 8 data channels, with the ability to expand to 32 channels, in groups of 8, as the need arises. Each data channel can transmit data at a nominal rate of 500,000 bits per second. The exchange can reach a peak rate of one word every 10 microseconds (100,000 words per second) transmitted to or received from core storage. The information rate of all input-output devices is controlled by their control units connected to the channels.

Read or write instructions are used to transfer data between the external units and core storage. To initiate an operation, a read or write instruction is issued by the stored program. This instruction specifies the external unit affected and the location of a control word. The control word defines the beginning of the storage area to be used for the data transfer and a count of the number of words to be transferred. These two pieces of information are sent to the exchange where they are held, and the computer proceeds to the next instruction in the program. When the desired input-output unit is available, it is started by the exchange, and reading or writing proceeds using the storage area indicated by the control word.

As each data word is transferred to or from storage, the control word in the exchange storage is modified. The data word address is advanced by one and the count is decreased by one. The control word in the exchange always contains the current data word address and a count of the number of words remaining to be transferred. The control word in the main storage is not affected during the data transfer; this control word retains the initial data word address and count and may be used repeatedly to transfer data to and from the same storage area.

The control word also contains a refill address that can specify the address of another control word. In this way, control words may be chained together to define storage areas that are not adjacent. Control words, thus, may be used first for reading, then for indexing while processing, and finally for writing the data from the same storage area.

All instructions for operating external units are issued by the computer program but are executed independently of the program. A number of data transfers can take place simultaneously, all sharing access to storage. The external device signals the program when the process is complete.

Because of the inflexible data rate from magnetic tapes, the exchange takes priority over the central processing unit in requesting core storage cycles. The over-all effect on processing is quite small, even with

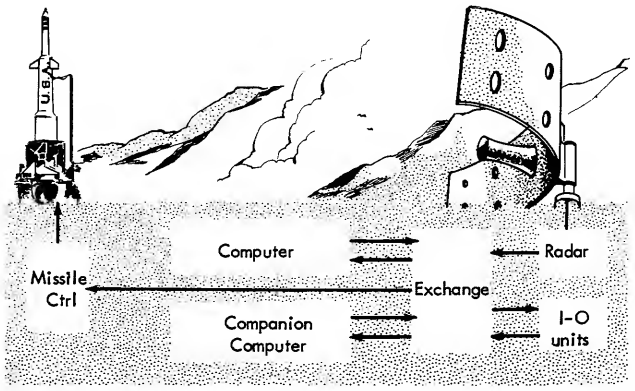


Figure 12. The Exchange in the Computer System

the exchange operating at peak rate (which is greater than the simultaneous reading rate of sixteen IBM 729 IV Magnetic Tape Units). For example, internal computing speed on a computer system with four units of core storage would be slowed less than 10 per cent with the exchange operating at maximum rate. As the number of core storage units decreases, computing speed is slowed somewhat more.

There are four basic variations of the reading or writing operations, depending on the setting of certain flag bits in the control word. These are:

Single Block Operation. The external unit may terminate reading or writing if it reaches the end of a block before the entire storage area defined by the control word has been used. A block of data is defined for each type of external unit as the amount of information recorded between adjacent starting and stopping points of the unit. The length of a block depends on the unit used and may be a card, a line of printing, or the data between two consecutive record gaps on magnetic tape.

Multiple Block Operation. More than one block may be read or written with one instruction, until the specified storage area is exhausted.

Chaining. After the storage area defined by the control word is exhausted, the exchange can substitute another control word and continue data transfer without stopping, using the storage area defined by the new control word.

Skipping. During reading, it is possible to suppress entry into storage of selected portions of the information being read.

Input-Output Units

Magnetic Tape Units

The IBM 729 IV Magnetic Tape Unit (Figure 13) is used with the 7030 system as intermediate or long term storage. The tape may be recorded at either of two character densities under program control. The density may also be changed by depressing a change density switch on the tape unit.

The higher density gives a maximum information rate of 62,500 six-bit bytes per second. This corresponds to 5864 64-bit words per second, or 170 microseconds per word. Tape speed is 112.5 inches per second.

The lower density gives an information rate of 22,500 six-bit bytes per second; tape speed is 112.5 inches per second. This density permits the system to handle magnetic tape that is fully compatible with other IBM data processing systems or auxiliary equipment operations such as off-line printers.

All tape units are connected to the exchange through a tape control unit. Up to eight tape units may be attached to one control unit, but only one of the eight can operate at one time. Simultaneous tape unit operation requires additional control units, and each control unit requires a different channel on the exchange. Tape units connected to different control units can operate independently of one another.

The six information tracks on the tape are labeled BA 8421. When these bits are placed in storage, the B bit is in the lowest-numbered, or left-most, bit position. Successive bytes are stored: BA 8421 BA 8421 B . . . in the order in which they are read. The tapes are normally operated with odd parity. Because some other systems use the same physical tapes with an even count parity method of checking, a mode of writing and reading even parity tapes is provided for communication purposes. Even parity tapes cannot be used for binary data.

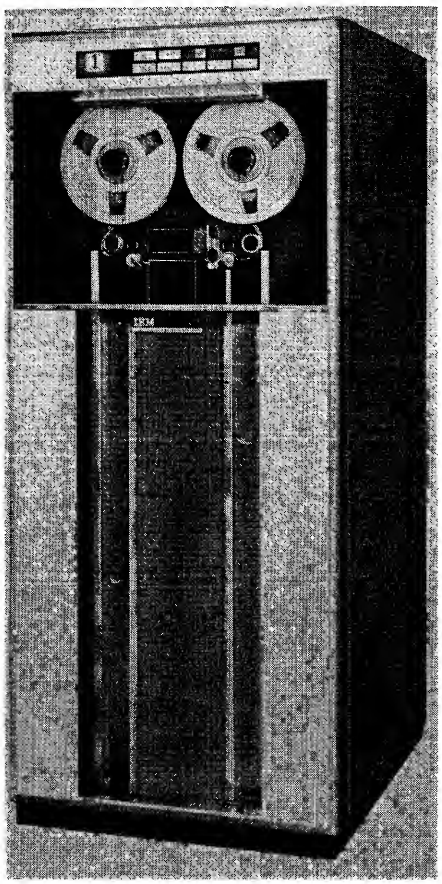


Figure 13. IBM 729 IV Magnetic Tape Unit

Card Reader

The IBM 7503 Card Reader (Figure 14) operates at 1000 cards per minute. Cards are loaded in the hopper face down, 9-edge first. They feed past a first reading station for checking, past a second reading station for data entry and to a drum type stacker. A clutch permits intermittent starting and stopping as far as the second reading station; from there the cards feed continuously to the stacker.

The card contents are transferred from the second reading station to a 960-bit core buffer in the reader control to change from a row-by-row representation to a column-by-column representation before entering the exchange and main storage. A complete 960-bit card image is thus transferred to storage, with a 1-bit representing a punch and a 0-bit representing no punch.

For greater simplicity and flexibility in data editing, the control panel is omitted from the card reader. The stored program has complete control of all editing functions.

When the card reader is initially loaded, the first card is read and the card image is stored in the buffer. A read instruction from the computer causes the buffer contents to be transferred to storage. As soon as the buffer is emptied, a new card cycle starts to place the next card in the buffer. Disagreements between the first and second reading stations cause a unit check indication to be given.

Card Punch

The IBM 7553 Card Punch (Figure 15) operates at 250 cards per minute. Cards are loaded in the hopper face down, 9-edge first. They feed past a punching station, past a reading station for checking, and to a drum type stacker. A blank station precedes the punching station so that, when the punch is loaded, there are two unpunched cards between the hopper and the punching station. A clutch permits intermittent starting and stopping as far as the reading

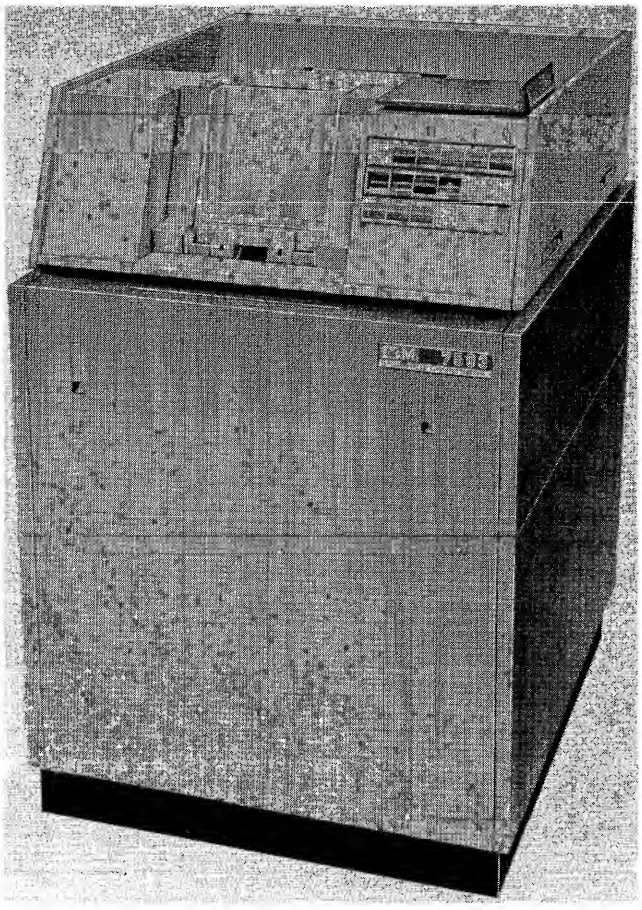


Figure 14. IBM 7503 Card Reader

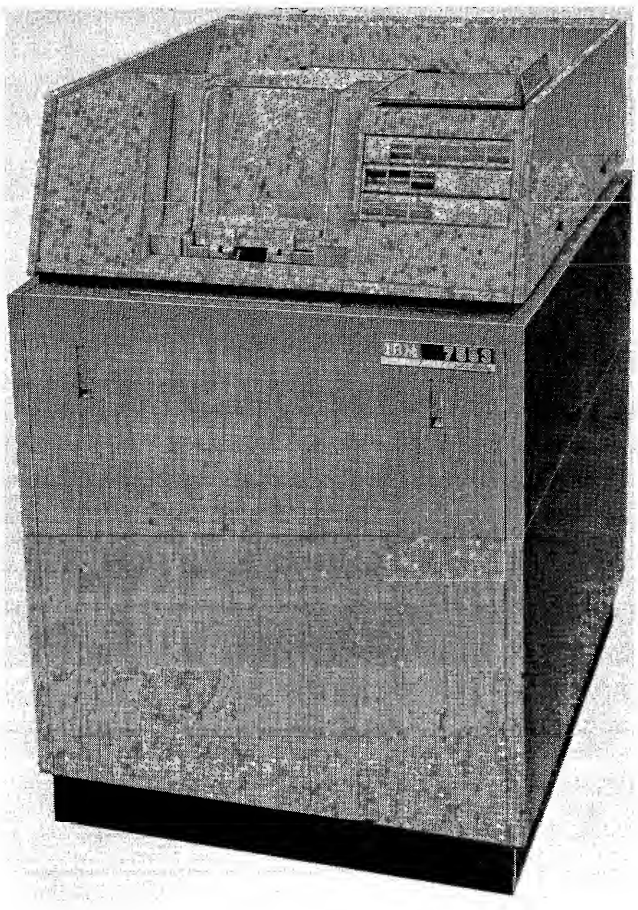


Figure 15. IBM 7553 Card Punch

station; from there the cards feed continuously to the stacker.

The information to be punched must be assembled in storage as a card image occupying up to 15 full words. The card image is transferred from storage to a 960-bit register in the punch control to change from the column-by-column representation to a row-by-row representation for punching. After the contents of the register are punched into a card, the card is read at the reading station to verify that the card contains the same punching as the buffer contents.

The control panel is omitted from the card punch for greater flexibility and, as with card reading, the stored program has full control of all data editing functions.

Each write instruction given to a card punch causes the buffer to be filled from storage. A card cycle is then started to punch the buffer contents into the card waiting just ahead of the punching station. An end of operation indication is not given until the end of the card cycle. At this time, the preceding card will have been checked at the reading station and a unit check signal given in case of an error.

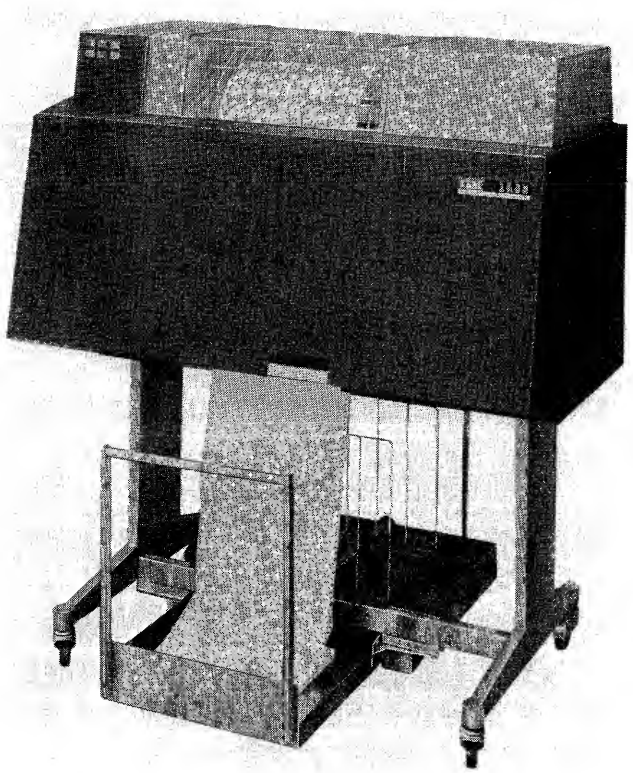


Figure 16. IBM 1403-2 Printer

Printer

The IBM 1403-2 Printer (Figure 16) is an electro-mechanical printer using engraved type. The alphabetic, numeric, and special characters are assembled in a chain (Figure 17). As the chain travels horizontally, each character is printed as it is positioned opposite a magnet-driven hammer which presses the form against the chain. Average speed is 600 lines per minute.

The printer unit consists of the chain printing mechanism, hydraulic carriage, paper and ribbon feeds, and magnetic drum. The control contains the control and interlock circuits and a buffer storage register capable of holding the data for one line of printing. Up to 132 positions may be printed on one line.

The chain printing mechanisms for the printer are in magazine form and are interchangeable to provide a wide choice of type fonts. The chains are composed of character sets called arrays. The standard chain has five arrays of 48 characters each.

The choice of type fonts might include Fortran type and mathematical or chemical symbol type. Even a limited Arabic language type could be used with the proper program control. All type fonts are quickly and readily interchangeable, giving a wider range of characters and symbols than ever before available on a data processing system printer. All information and data to be printed must be coded in an 8-bit code before being sent to the printer and will be printed

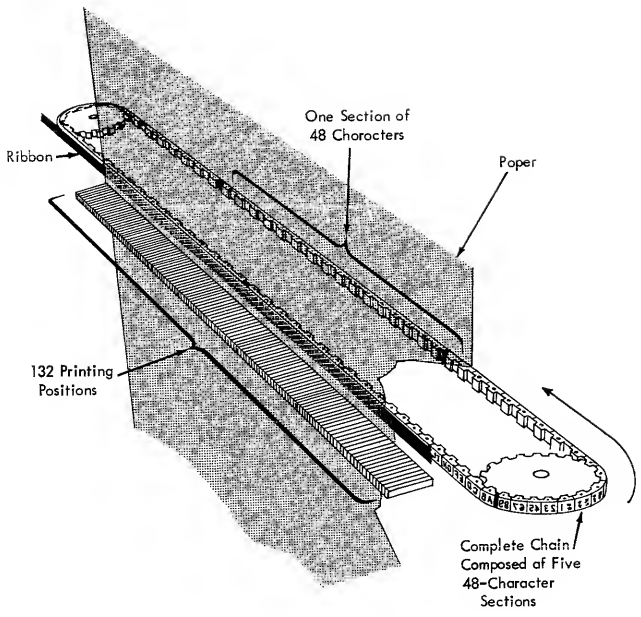


Figure 17. Printer Type Chain

with one line containing up to 17 words. A write instruction then transfers this block of words from storage to the print buffer, which stores one complete line of printing at a time. All characters printed are checked at the time of printing. Other than loading the carriage with the proper forms and control tape, no operator set-up time is required. The stored program has full control over the printing operation for greater flexibility; there is no control panel.

High-Speed Magnetic Disk Storage

High-speed magnetic disk storage provides large external capacity to supplement internal core storage. Each disk unit can contain 2,097,152 full words. Data may be transmitted between disk storage and internal storage at an average rate of one full word every eight microseconds. It is possible to address up to 32 disk units.

A high-speed disk storage system consists of one IBM 7303 Disk Storage and its associated IBM 7612 Disk Synchronizer. Within the disk unit, the total storage area is divided into 4096 addressable locations called arcs. An arc contains 512 words and is the smallest addressable portion of disk storage. Eight arcs form a track which is identified by the nine high-order bits of the arc address.

A disk storage unit consists of 39 magnetically coated disks for data storage. Both sides of each disk are used for recording, making a total of 78 disk faces. Each face has its own access arm (read-write head). One set of 39 disk faces covers all even numbered tracks while the other set covers all odd numbered ones. Thus, two consecutively numbered tracks are never located on the same set of disk faces. This arrangement enables one set of heads to be positioned and ready to read or write while the previous set of heads are reading or writing.

Console

The IBM 7152 Operator's Console is separate from the main computer and is an optional input-output device. The keys, lights, and digital display (Figure 18); two sets of program switches (Figure 19); and the console printer are all subject to programmed interpretation and control. An IBM 7623 Console Control is used for both the console and the console printer. The interpretive program may give these devices sophisticated control functions or may ignore them. This interpretive approach to the console gives exceptional flexibility and makes possible console facilities that remain adequate as new operating techniques are developed.

The console contains a printer for input and output, a number of toggle switches and columnar switches for input, and lights and digital displays for output. Also available are three variable controls to permit the entry of analog information. The switches are scanned on receipt of a read instruction from the computer and their content stored in core storage.

Various customer engineering control consoles are used throughout the system to assist in localizing malfunctions and program errors. These consoles are widely used during diagnostic programming sessions to locate questionable machine components before they become troublesome. Error indicators are also under constant monitoring, and a recording is made of all error conditions.

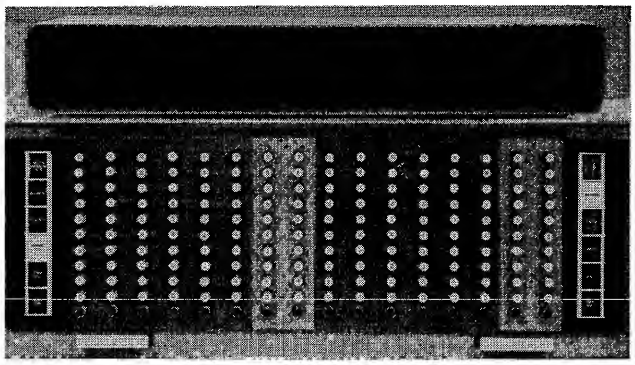


Figure 18. Keys, Lights, and Digital Display

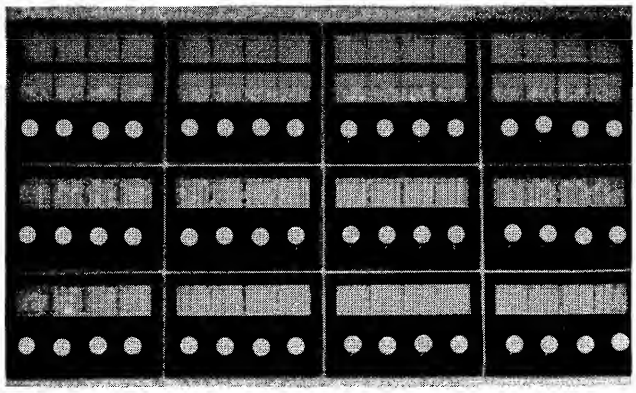


Figure 19. Program Switches, Left Side

System Reliability and Accuracy

The power and speed of the IBM 7030 system is derived from a balanced system design in which every component operates at its optimum rate. Important facets of the 7030 design are reliability and self-checking. Solid state design results in increased reliability, a lower cost system, and decreased maintenance requirements. To insure circuit reliability, all electronic components operate at a level below rated capacity.

The physical arrangement of system components also offers an important benefit in operating efficiency. Components requiring operator attention may be situated for accessibility and convenience. Control and arithmetic components are consolidated into a single set of modular cabinets which may be located less prominently.

The self-checking features built into the 7030 are designed to insure the maximum degree of error detection with minimum operator interruption. The use of error check and correct (ECC) bits permits automatic correction of any single bit errors within a given word without loss of computer time or expenditure of programmer effort. Double bit errors are detected and indicated to the operator.

The transmission of data, on any move through the system, is continually checked for validity. Three types of data checking are used in the 7030 computer:

1. Error Check and Correct.
2. Parity.
3. Residue.

At least one of these methods is used with any given data transmission, and in some cases, a combination of methods is used.

ECC checking is performed on all data transfers between core storage and the exchange, and between core storage and the central processing unit.

Parity checking is used to check data transfers which do not involve the main core storage unit. This method is basically a computed check in which an original parity, carried with the data, is compared against a computed parity and then checked for validity.

Floating point operations are checked on an overall basis by residue checking. A computed method is used, as with parity checking, but a different code is used.

Duplicate circuitry is also used as a checking method in the 7030. The entire serial arithmetic unit is composed of duplicate circuits to further increase the reliability of the system.

Appendix

Special Storage Location Assignments

LOCATION	NAME	LENGTH	BIT ADDRESS
0	Zero	64	0–63
1 (P, a)	Interval Timer	19	0–18
1 (P, b)	Time Clock	36	28–63
2 (P)	Interruption Address	18	0–17
3 (P)	Upper Boundary	18	0–17
3 (P)	Lower Boundary	18	32–49
3 (P)	Boundary Control Bit	1	57
4	Maintenance Bits	64	0–63
5 (b)	Channel Address	7	12–18
6	Other Central Processing Unit	19	0–18
7	Left Zeros Count	7	17–23
7	All Ones Count	7	44–50
8	Left Half of Accumulator Register	64	0–63
9	Right Half of Accumulator Register	64	0–63
10	Accumulator Sign Byte	8	0–7
11 (c)	Indicator Register	64	0–63
12 (d)	Mask Register	64	0–63
13	Remainder Register	64	0–63
14	Factor Register	64	0–63
15	Transit Register	64	0–63
16–31	Index Registers 0–15	64	0–63

NOTES:

P = Permanently protected area of storage.

a = Read–only; except for Store Value, Store Count, Store Refill, and Store Address instructions.

b = Read–only.

c = Bit positions 0–19 are read–only.

d = Bit positions 0–19 are always ones, and positions 48–63 are always zeros.

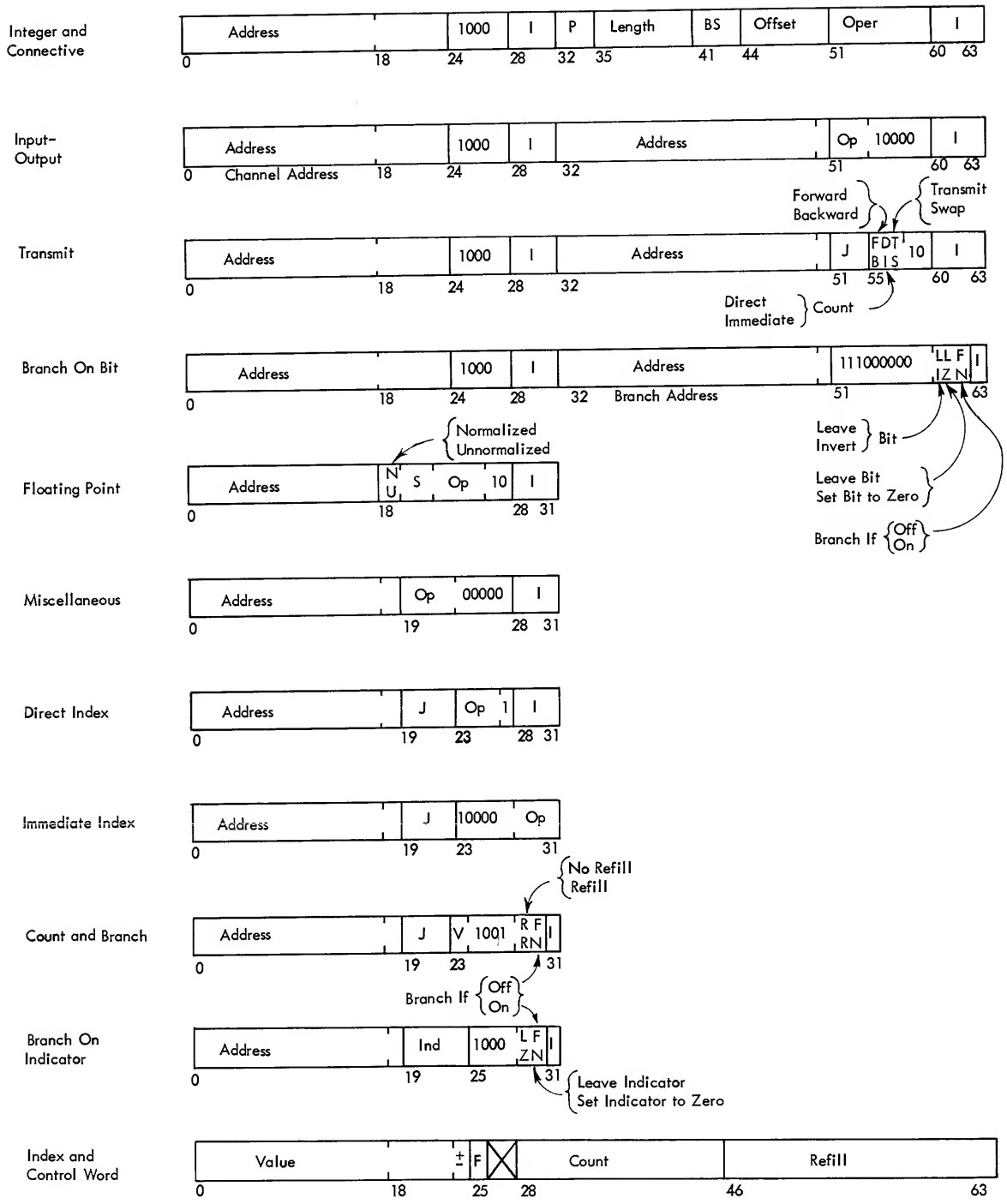
(Read–only means that these positions cannot be written into.)

Alphabetic Listing of Mnemonic Operation Codes

+	Add
+MG	Add to magnitude
*	Multiply
*+	Add product
/	Divide
B	Branch
BB	Branch on bit
BD	Branch disabled
BE	Branch enabled
BEW	Branch enabled and wait
BI	Branch on indicator
BR	Branch relative
C	Connect
C+	Add to count
CB	Count and branch
CBR	Count, branch, and refill
CCW	Copy control word
CM	Connect to storage
CV	Convert
CT	Connect for test
CTL	Control
D+	Add double
D+MG	Add double to magnitude
D*	Multiply double
D/	Divide double
DCV	Convert double
DL	Load double
DLWF	Load double with flag
E+	Add to exponent
EX	Execute
EXC	Execute indirect and count
F+	Add to fraction
K	Compare
KC	Compare count

KE	Compare if equal
KF	Compare field
KFE	Compare field if equal
KFR	Compare field for range
KMG	Compare magnitude
KMGR	Compare magnitude for range
KR	Compare for range
KV	Compare value
L	Load
LC	Load count
LCV	Load converted
LFT	Load factor
LOC	Locate
LR	Load refill
LTRCV	Load transit converted
LTRS	Load transit and set
LV	Load value
LVE	Load value effective
LVS	Load value with sum
LWF	Load with flag
LX	Load index
M+	Add to storage
M+MG	Add magnitude to storage
M+1	Add one to storage
NOP	No operation
R	Refill
R/	Reciprocal divide
RCZ	Refill on count zero
RD	Read
REL	Release
RNX	Rename
SC	Store count
SHF	Shift fraction
SIC	Store instruction counter if
SLO	Store low order
SR	Store refill
SRD	Store rounded
SRT	Store root
ST	Store
SV	Store value
SVA	Store value in address
SWAP	Swap
SX	Store index
T	Transmit
V+	Add to value
V+C	Add to value and count
V+CR	Add to value, count and refill
W	Write
Z	Store zero

IBM 7030 Instruction Formats



IBM

International Business Machines Corporation
Data Processing Division, 112 East Post Road, White Plains, N.Y.

Printed in U.S.A. D22-6513